基于最近邻距离权重的 ML-KNN 算法 *

陆 凯，徐 华+

(江南大学 物联网工程学院, 江苏 无锡 214122)

摘 要：在大数据环境下，K 近邻多标签算法（ML-KNN）高时间复杂度的问题显的尤为突出；此外，ML-KNN 也没有考虑 k 个近邻对最终分类结果的影响。针对上述问题进行研究，首先将训练集进行聚类，再为测试集找到一个距离其最近的训练数据簇作为新的训练数据集；然后计算最近邻样本的距离权重，并用该权重描述最近邻和其他近邻对预测结果的影响；最后使用新的目标函数为待测样本分类。通过在图片、Web 页面文本数据等数据集上的实验表明，所提算法得到了更好的分类结果，并且大大降低了时间复杂度。

关键词：多标签分类；ML-KNN；聚类；最近邻；距离权重

中图分类号：TP301.6 doi: 10.19734/j.issn.1001-3695.2018.09.0738

ML-KNN algorithm based on nearest neighbor distance weight

Lu Kai, Xu Hua+

(*School of Internet of Things Engineering*, *Jiangnan University*, *Wuxi Jiangsu* 214122, *China*)

Abstract: In order to efficiently apply multi-label K-nearest neighbor(ML-KNN) in big dataset, this paper first conducted clustering algorithm to separate the training dataset into several parts, from which found a nearest cluster as new training set for test dataset; and then calculated nearest neighbor distance weight, by which the effect of k neighbors was described. Finally, an unseen sample could be classified by this new method. Numberical simulation results in different datesets show that a better classification result is obtained, and the time complexity of the algorithm is also improved.

Key words: multi-label classification; ML-KNN; cluster; nearest neighbor; distance weight

0 引言

互联网的发展直接造成数据规模和数据信息量的飞速增长，这就导致了传统的单标签分类已经无法适应当下的分类需求，因此多标签分类逐渐成为数据分类的重点话题。在传统的单标签分类[1]中，每一个训练样本都只对应着一个标签 l，其中 l 来自于一个有限的、互斥的标签集合 L。假设 $D=\{(m_1,l_1),(m_2,l_2),\ldots,(m_n,l_n)\}$，其中 m 表示输入的数据，l 表示 m 从属的单标签。如果标签的数量为 2，则称为二分类，而如果标签数量大于 2，则称为多标签分类。多标签（multi-label）分类[2]和多类别（multi-class）分类[3]是完全不同的两个概念。多类别分类是在两个以上的类中预测出属于样本的一个类，所以其本质还是单标签分类。

一般来说，多标签分类方法主要可以分为问题转换法、算法适应法和集成方法[4,5]三类。问题转换法是将一个多标签分类问题转换为一个或者多个单标签分类问题，其中应用最为广泛方法的则是将每一个标签的预测都看做是一个独立的二分类任务，即：对于每一个不同的标签 $\lambda\in L$ 都给出一个二分类器 h_λ： $X\rightarrow\{\neg\lambda,\lambda\}$；然后将原始数据集转换为 $|L|$ 个数据集 D_λ，其中 λ 表示原始数据包含该标签，而 $\neg\lambda$ 则表示原始数据不包含该标签，这种方法被称为二分关联（BR）算法[6,7]。算法适应法是将一些传统的分类器扩展为一个可以直接处理多标签分类问题的多标签分类器，多标签 K 近邻算法（ML-KNN）就是该方法比较典型例子。集成方法则是将多个一种或多种多标签基分类器向融合以达到更好的分类效果。

近年来涌现出不少有关 ML-KNN 算法的研究，并都取得了不错的成果。文献[8]通过在 ML-KNN 算法中引入辅助标签的方式提出了一种分层的 ML-KNN 方法，引入辅助标签的目的就是弥补了 ML-KNN 算法没有考虑标签之间具有相关性的缺陷，从而改善了算法的最终结果；在文献[9]中，首先使用原型选择算法（PS）来减少训练数据的数量，并且在这个删减过的数据集上，根据标签之间的关联程度来得到一个标签的排序，通过这种方法可以将很多不相关的原型丢弃；文献[10]则是先利用线性判别式分析（LDA）提取出数据的特征，再降低整个数据矩阵的维度，从而训练得到一个分类器和标签之间的相关性，最后可获得到分类结果；在文献[11]中，首先计算出标签集合中每对标签间的条件概率，然后对于即将被预测的标签，将其与已经预测的标签间的条件概率进行排序，求出最大值，最后将最大值与对应标签值相乘同时结合最大化后验概率来构造多标签分类模型。从以上的几种研究情况来看，现在大部分研究都集中于多标签的标签相关性上，如文献[12]也是一个利用标签相关性来提升 ML-KNN 算法的研究方式。标签相关性确实是提升了算法的运行结果，但是当数据量很大时，也是存在标签量很大的情况，这必然会造成更大的时间消耗。另外，说到大数据自然会想到 Hadoop、Spark 等分布式技术，文献[13]就是结合 Spark 的优越性和 ML-KNN 算法的特性提出了基于 Spark 框架下的多标签最近邻算法。而在实际的生产生活中，Spark 需要的分布式环境的成本也是不可忽视的。在本文所提的算法中，利用聚类算法的同时引入最近邻距离权重就能达到降低算法时间复杂度和提升算法性能的双重目的。

收稿日期：2018-09-19；**修回日期：**2018-11-08 **基金项目：**国家教育部—新华三集团“云数融合”基金项目（2017A13055）

作者简介：陆凯（1994-），江苏盐城人，硕士，主要研究方向为大数据及机器学习；徐华（1978-），女（通信作者），江苏无锡人，副教授，硕导，博士，主要研究方向为计算智能、车间调度、大数据等（joanxh2003@163.com）．

1 相关介绍

1.1 多标签分类

多标签分类问题可以用数学语言描述为：

给定一个 d 维样本空间 m，有限的标签集合 $L=\{l_1,l_2,...,l_q\}$；那么训练集则可以表示为 $D=\{(m_1,L_1),(m_2,L_2),...,(m_n,L_n)\}(m_i\in R^d,L_i\subseteq L)$，需要计算出一个多标签分类器 $h:D\times L\to R$。

1.2 ML-KNN 算法简述

ML-KNN[14]是基于传统的 KNN 算法和最大后验概率法则的懒惰型多标签学习算法，它的主要思想是一个样本的标签集合可以由该样本的近邻来决定。给定一个待测样本 x，假设它的标签集合为 $L(x)$，ML-KNN 算法首先在训练数据集中找出 x 的 k 个近邻，再统计这些近邻中属于每一个标签的数量，记为 j。然后根据最大后验概率准则来确定测试样本的标签集合，且每一个 $l_i\in L$ 的后验概率计算如下：

$$P(l_i\in L(x)\,|\,E_j^{l_i})=P(l_i\in L(x))P(E_j^{l_i}\,|\,l_i\in L(x)) \tag{1}$$

最后，对于每一个 $l_i\in L$ 是否在 x 的标签集合中，ML-KNN 算法使用如下规则：

$$y_x(l_i)=\begin{cases}1, P(l_i\in L(x)\,|\,E_j^{l_i})\ge 0.5\\ 0, P(l_i\in L(x)\,|\,E_j^{l_i})<0.5\end{cases} \tag{2}$$

其中：$E_j^{l_i}$ 表示 x 的 k 个近邻中有 j 个样本属于标签 l_i 的事件；

如果 $y_x(l_i)=1$ 则表示 l_i 在 x 的真实标签集合中，如果 $y_x(l_i)=0$ 则表示 l_i 不在 x 的真实标签集合中。

2 改进算法

对于数据集中的每一个样本，它的 k 个近邻的标签集合理论上应该跟它自身的标签集合有一定程度上的相似，这种相似度会随着近邻到样本距离的改变而改变，并且距离大，相似度越小。而 ML-KNN 算法的计算过程中并没有将这种关系考虑进去，针对这一缺陷，本文同时考虑了待测样本的 k 个近邻和最近邻的影响，并利用权重来衡量这种影响。因此根据式(1)可得到一个新的分类函数：

$$P(l_i\in L(x)\,|\,E_j^{l_i})=w*NN_x(l_i)+(1-w)*P(l_i\in L(x))P(E_j^{l_i}\,|\,l_i\in L(x)) \tag{3}$$

其中：w 表示由 x 的最近邻到 x 的距离所转换而来的权重，也是 x 的最近邻在本文改进算法中的权重；$1-w$ 则表示 x 的 k 个近邻的权重；$NN_x(l_i)$ 表示 x 的最近邻样本是否含有 l_i 标签，取值只能是 0 或者 1。

由式(3)可知，本文算法的关键是如何确定权重 w，将距离转换为权重主要有反函数、减函数、高斯函数三种方法。对于一个距离 d，反函数最为简单，将距离加上一个常量之后取倒数，即 $w=\frac{1}{d+c}$，加入常量 c 的目的是为了避免当距离接近 0 时而导致反函数为无穷大，因此反函数的缺点也比较明显，它会赋予近邻项很大的权重，而随着 d 的增大 w 则会以一个极快的速度递减；减函数跟反函数类似，也是引入了一个常量 c，若 $d\le c$，则 $w=c-d$，否则 $w=0$，虽然该函数克服了近邻分配权重过大的问题，但是 w 一定会慢慢递减到 0；高斯函数则是利用式(4)将距离转换成权重：

$$w=a*e^{-\frac{d^2}{2c^2}} \tag{4}$$

其中：a 和 c 是常数；d 表示的距离都是欧氏距离。高斯函数克服了近邻项分配权重过大的潜在问题，并且权重会随着距离的增加而减少，与减函数不同的是这里的权重始终不会跌至 0，所以本文将采用高斯函数的方法将距离转换为权重。本文改进算法的详细步骤如下：

算法 1：改进 ML-KNN 算法的详细步骤

输入：训练数据，测试数据。

输出：分类结果。

1 利用 k-means 聚类算法将训练数据分为 r 个聚类中心，记为 $R_1,R_2,...,R_r$；

2 利用 k-means 聚类算法将测试数据分为 t 个聚类中心，记为 $T_1,T_2,...,T_r$；

3 当 $i\in\{1,2,...,t\}$ 时：

4 计算 T_i 到 R_j 的欧式距离，记为 $D(T_i,R_j)$，j=1,2,...,r；

5 根据第 4 步获取距离 T_i 最近的 R_j，

$R_j=\min\{D(T_i,R_j)\}$,j=1,2,...,r;

6 将 T_i 对应的簇作为新的测试数据集，记为 $NewY$；

7 将第 5 步得到的 R_j 对应的簇作为新的训练数据，记为 $NewX$；

8 对于每一个 $x_u\in NewX$，计算每一个 $l_i\in L$ 的先验概率：

$$P(l_i\in L(x_u))=(s+\sum_{u=1}^{m}y_{x_u}(l_i))/(s*2+m)$$

$$P(l_i\notin L(x_u))=1-P(l_i\in L(x_u))$$

9 获取 x_u 的 k 个近邻 N_{x_u}、最近邻 NN_{x_u} 以及最近邻距离 x_u 的距离 d，并使用式(4)将其转换为 w

10 计算 $l_i\in L$ 的后验概率：

令 $p\in\{0,1,...,k\}$

$$P(E_p^{l_i}\,|\,l_i\in L(x_u))=(s+c[p])/((s*|N_{x_u}|+1)+\sum_{q=0}^{k}c[q])$$

$$P(E_p^{l_i}\,|\,l_i\notin L(x_u))=(s+c'[p])/((s*|N_{x_u}|+1)+\sum_{q=0}^{k}c'[q])$$

11 利用式(3)计算每一个 $z_v\in NewY$ 拥有 l_i 的后验概率，得到最终预测结果

根据算法 1，本文先是使用 K-means 分别将训练数据和测试数据聚类[15]，再为每一个测试数据簇选择一个距离它最近的训练数据簇，这样一方面减少了数据的规模，从而减少了 ML-KNN 算法的计算量，最后达到降低算法时间复杂度的目的；另一方面距离测试数据最近的训练簇中的数据相较于其他训练数据和测试数据具有一定程度上相似度，从而也提高了多标签算法的各种评价指标。然后再使用本文改进 ML-KNN 算法对数据分类。

3 实验结果与分析

为了方便比较算法的性能，将本文提出的改进算法称为 DML-KNN，文献[16]中提出的改进算法称为 IML-KNN，并分别在 MATLAB 上实现。MATLAB 是一个功能强大且使用的算法仿真工具。

3.1 评价指标

a)Hamming loss。计算一个样本标签对被错误分类的次数，比如一个本该属于样本的标签没有被预测到，或者一个不属于样本的标签被预测到了。

$$hloss(h)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{Q}|h(x_i)\Delta Y_i| \tag{5}$$

其中：Q 是标签数量；$h(x_i)$ 表示分类结果；$h(x_i)\Delta Y_i$ 表示样本真实标签集合与分类结果的对称差。

b)Coverage。评估为了涵盖所有可能的样本标签需要平均给出的标签列表。

$$\mathrm{cov}(f)=\frac{1}{m}\sum_{i=1}^{m}\max_{y\in Y_i} rank_f(x_i,y)-1 \tag{6}$$

c)One-error。评估排在前面的标签不在样本真实标签中的次数。

$$one-error(f)=\frac{1}{m}\sum_{i=1}^{m}H(x_i) \tag{7}$$

若排在首位的标签在样本真实标签中，则 $H(x_i)=1$，否则 $H(x_i)=0$。

d)Ranking loss。评估被反向排序的标签对的平均分。

$$rloss(f)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{|Y_i||\overline{Y_i}|} \tag{8}$$

其中：$\overline{Y_i}$ 是 Y_i 在总标签集合上的补集。

e)Average precision。该指标的含义与 One-error 相反，它评估的是排在前面的标签在样本真实标签中的次数。

$$avgp(f)=\frac{1}{m}\sum_{i=1}^{m}\frac{\sum_{y\in Y_i}\frac{|\{y'\mid rank_f(x_i,y')\le rank_f(x_i,y'),y'\in Y_i\}|}{rank_f(x_i,y)}}{|Y_i|} \tag{9}$$

以上五个评价中，前四个越小越好，最后一个越大越好。

3.2　数据集

表 1 中记录的是选取于 keel 的四个数据集的详细信息，它们分别为电子邮件信息数据集 enron、图片数据集 scene、酵母细胞信息数据集 Yeast 和包含 Web 页面文本数据的数据集 delicious。其中标签基数表示一个样本平均具有几个标签，标签密度是由标签基数除以标签总数。

表 1　实验数据信息

Table 1　Experimental data information

名称	enron	scene	Yeast	delicious
样本数	1702	2407	2417	16105
标签数	53	6	14	983
标签基数	3.3784	1.074	4.237	19.02
标签密度	0.064	0.179	0.303	0.019
训练数	1532	1927	2177	14495
测试数	170	480	240	1600

3.3　实验结果及分析

基于 3.2 节中的四个数据集和 3.1 节中的五个评价指标，分别进行本文算法的性能评估。由算法 1 可知，本文算法具有以下几个关键参数：最近邻数 k、高斯函数中的两个可调参数 a,c、训练数据聚类簇数 r、测试数据聚类簇数 t。对于 ML-KNN 算法，最近邻数 k 应当避免过小或者过大，因为过小时，近邻中包含的信息也会越少；而过大时，虽然近邻中包含的信息量大了，但是这会很大程度上增加算法的计算量。结果多次实验，本文将 k 的值定为 20。表 2~5 分别是当 a,c 不同时，三种算法在四个数据集上的实验结果。

表 2　三种算法在数据集 enron 上的实验结果

Table 2　Results of three algorithms on enron

			a=5	a=7	a=10
			c=1/2	c=1/3	c=1/2
指标	ML-KNN	IML-KNN	DML-KNN		
hloss	0.0507	0.0502	0.0453	0.0453	0.0461
rloss	0.1002	0.0896	0.0616	0.0595	0.0581
one-error	0.3	0.288	0.1534	0.1503	0.1534
coverage	14.365	13.306	10.057	9.8392	9.8032
avgprec	0.638	0.647	0.7421	0.7439	0.7514
Time(s)	20.781	20.53	18.976	18.895	19.153

表 3　三种算法在数据集 scene 上的实验结果

Table 3　Results of three algorithms on scene

			a=5	a=6	a=8
			c=1/3	c=1/2	c=1/5
指标	ML-KNN	IML-KNN	DML-KNN		
hloss	0.0906	0.083681	0.0697	0.0723	0.0696
rloss	0.0723	0.069687	0.0667	0.0692	0.0657
one-error	0.2291	0.22292	0.1861	0.1973	0.1815
coverage	0.4542	0.43958	0.40688	0.4294	0.4012
avgprec	0.867	0.87111	0.88854	0.8819	0.891
Time(s)	14.53	14.3504	6.1741	5.9862	6.065

表 4　三种算法在数据集 Yeast 上的实验结果

Table 4　Results of three algorithms on Yeast

			a=5	a=7	a=10
			c=1/2	c=1/4	c=1/6
指标	ML-KNN	IML-KNN	DML-KNN		
hloss	0.20119	0.20179	0.24084	0.2022	0.2012
rloss	0.17932	0.17843	0.19358	0.1857	0.1755
one-error	0.24167	0.2333	0.26088	0.2419	0.21385
coverage	6.4333	6.425	6.577	6.5398	6.4809
avgprec	0.75232	0.75351	0.73327	0.7464	0.76022
Time(s)	5.435	5.4848	4.083	4.1248	4.0908

表 5　三种算法在数据集 delicious 上的实验结果

Table 5　Results of three algorithms on delicious

			a=5	a=8	a=10
			c=1/2	c=1/3	c=1/5
指标	ML-KNN	IML-KNN	DML-KNN		
hloss	0.01851	0.018468	0.0143	0.016	0.0133
rloss	0.12416	0.11953	0.1063	0.115	0.0887
one-error	0.39006	0.38199	0.275	0.259	0.2164
coverage	578.954	556.0199	378.1	472.45	346.48
avgprec	0.32995	0.34044	0.4982	0.4195	0.5007
Time(s)	899.5247	916.6451	343.14	462.93	343.81

由上面的四个表格可知，在数据集 enron 上，当 a=10,c=1/2 时本文算法效果最好；在数据集 scene 上，当 a=8,c=1/5 时本文算法效果最好；在数据集 Yeast 上，当 a=10,c=1/6 时本文算法效果最好；在数据集 delicious 上，当 a=10,c=1/5 时本文算法效果最好；同时也不难看出，本文算法在时间复杂度上也具有不小的优势，尤其在数据集 delicious 上，当数据量大幅度增加时，本文算法在五大性能指标和时间复杂度上都有大幅度的改进。进一步地，由算法 1 可知，不同的聚类簇数对本文算法也有至关重要的影响，因为它直接决定了测试数据对应的训练集，从而最终影响到预测结果。当 a、c 确定时，表 6~9 反映了当聚类簇数不同时，本文算法在四个数据集上的性能。

表 6　数据集 enron 上不同聚类簇数的影响

Table 6　Effect of different r,t on enron

	r=2，t=2	r=3，t=2	r=3，t=3
hloss	0.046087	0.052775	0.055204
rloss	0.058146	0.10065	0.058409
one-error	0.15337	0.29707	0.27053
coverage	9.8032	12.7693	9.9893
avgprec	0.75137	0.6572	0.71208
Time(s)	19.1834	17.421	17.2456

表 7　数据集 scene 上不同聚类簇数的影响

Table 7　Effect of different r,t on scene

	r=2，t=2	r=3，t=3	r=4，t=3
hloss	0.069413	0.072451	0.11154
rloss	0.065248	0.069544	0.096498
one-error	0.18011	0.19142	0.31161
coverage	0.39939	0.43201	0.56425
avgprec	0.89178	0.88367	0.82217
Time(s)	6.0315	4.4674	4.0325

表 8　数据集 Yeast 上不同聚类簇数的影响

Table 8　Effect of different r,t on Yeast

	r=2，t=2	r=3，t=2	r=3，t=3
hloss	0.20412	0.21709	0.21794
rloss	0.17922	0.19141	0.18643
one-error	0.20681	0.26982	0.25124
coverage	6.5013	6.6214	6.6703
avgprec	0.75848	0.7398	0.74164
Time(s)	4.1438	2.2736	2.926

表 9　数据集 delicious 上不同聚类簇数的影响

Table 9　Effect of different r,t on delicious

	r=11，t=4	r=13，t=3	r=14，t=4
hloss	0.014148	0.015778	0.013979
rloss	0.10453	0.13485	0.10342
one-error	0.28109	0.32058	0.31589
coverage	378.5584	451.5746	373.349
avgprec	0.49485	0.41648	0.47827
Time(s)	416.7709	419.4521	422.1061

由表 6~9 可知，在数据集 enron、scene、Yeast 上，r=2，t=2 时算法效果最好；在数据集 delicious 上，当 r=11，t=4 时算法效果最好。与此同时，r、t 的取值需要适当，若 r 太小，则起不到降低时间复杂度的效果，但当 r 太大时，虽然时间可能会减少，但是也可能会大幅降低算法的精度；对于 t 来说，它决定了算法的循环次数，即决定了算法的计算量，所以 t 的取值不宜太大。图 1~6 直观地体现了本文算法的优越性。其中 coverage 和时间会因为数量规模的不同而出现较大的差异，所以本文在这两者的比较上利用百分比的方式。

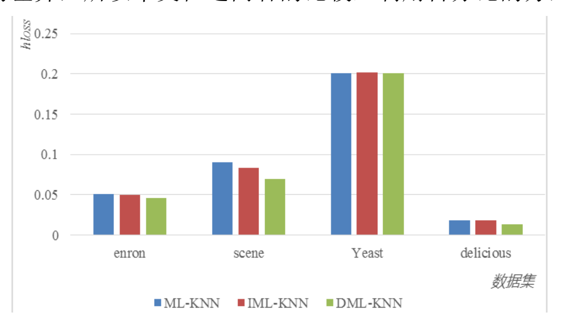


图 1　三种算法的 hloss 对比

Fig. 1　Hloss comparison of three algorithms

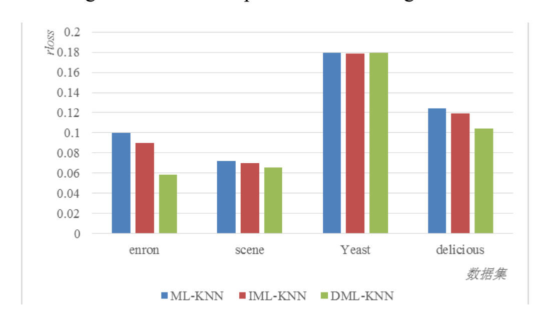


图 2　三种算法的 rloss 对比

Fig. 2　Rloss comparison of three algorithms

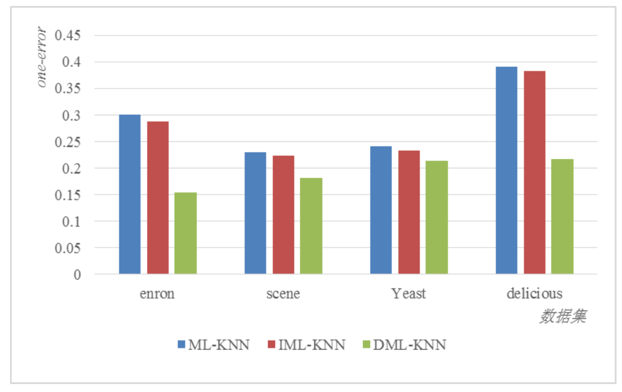


图 3　三种算法的 one-error 对比

Fig. 3　One-error comparison of three algorithms

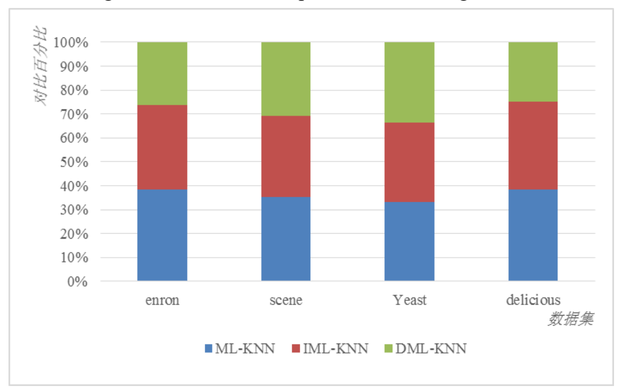


图 4　三种算法的 coverage 对比

Fig. 4　Coverage comparison of three algorithms

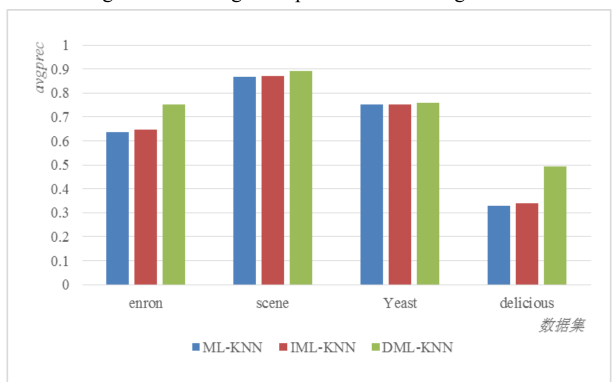


图 5　三种算法的 avgprec 对比

Fig. 5　Avgprec comparison of three algorithms

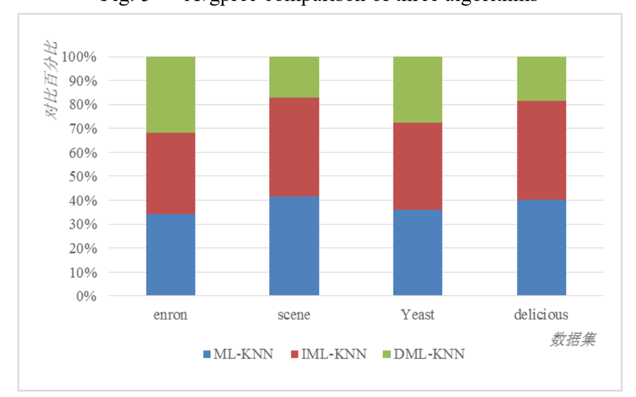


图 6　三种算法的时间对比

Fig. 6　Time comparison of three algorithms

4　结束语

针对传统的 ML-KNN 算法高时间复杂度的问题，本文先将训练数据和测试聚类以降低数据的规模，从而减少算法的

计算量，最终达到降低时间复杂度的目的；同时本文还考虑了 k 个近邻会因为距离的不同而对预测结果产生不同的影响，并将这种影响利用最近邻距离权重衡量。结合聚类和最近邻距离权重，本文提出了一种改进的 ML-KNN 算法。实验结果也表明本文算法会取得更好的分类效果，并且在多标签评价指标上也优于原始算法。

参考文献：

[1] Liu Chunming, Cao Longbing. A coupled K-nearest neighbor algorithm for multi-label classification [C]// Advances in Knowledge Discovery and Data Mining. Berlin: Springer International Publishing, 2015: 176-187.

[2] Chen Zijie, Hao Zhifeng. Latent semantic KNN algorithm for multi-label learning [C]// Proc of International Conference on Machine Learning and Cybernetics. Berlin: Springer Berlin Heidelberg, 2015: 278-284.

[3] Zhou Ligang, Tam Kwoping, Fu Jita. Predicting the listing status of Chinese listed companies with multi-class classification models [J]. Information Sciences, 2016, 328 (C): 222-236.

[4] Mahdavi-Shahri A, Houshmand M, Yaghoobi M, et al. Applying an ensemble learning method for improving multi-label classification performance [C]// Proc of Signal Processing and Intelligent Systems. Berlin: Springer Berlin Heidelberg, 2017: 14-15.

[5] 李玲，刘华文，马宗杰，等. 基于特征选择的集成多标签分类算法 [J]. 计算机工程与科学, 2013, 35 (10): 137-143. (Li Lin, Liu Huawen, Ma Zongjie, et al. An ensemble multi-label classification method using feature selection [J]. Computer Engineering & Science, 2013, 35 (10): 137-143.)

[6] Tanaka E A, Macedo A A. A multi-label approach using binary relevance and decision trees applied to functional genomics [J]. Journal of Biomedical Informatics, 2015, 54 (C): 85-95.

[7] Montañes E , Senge R , Barranquero J , et al. Dependent binary relevance models for multi-label classification [J]. Pattern Recognition, 2014, 47 (3): 1494-1508.

[8] Qi Hongwei, Zhou Yanquan, Guo Qi. A hierarchical ML-KNN method for complex emotion analysis on customer reviews [C]// Proc of International Conference on Mechatronics Engineering and Information Technology. 2016: 6.

[9] Calvo-Zaragoza J, Valero-Mas J J, Rico-Juan J R. Improving KNN multi-label classification in Prototype selection scenarios using class proposals [J]. Pattern Recognition, 2015, 48 (5): 1608-1622.

[10] Li Zhiqiang. An improved ML-KNN multi-label classification model based on feature dimensionality reduction [C]// Proc of International Conference on Computer, Mechatronics and Electronic Engineering. [S.l.]: Science and Engineering Research Center , 2016: 4.

[11] 檀何凤，刘政怡. 基于标签相关性的 K 近邻多标签分类方法 [J]. 计算机应用，2015, 35 (10): 2761-2765. (Tan Hefeng, Liu Zhengyi. Multi-label K-nearest neighbor algorithm by exploiting label correlation [J]. Journal of Computer Applications, 2015, 35 (10): 2761-2765.)

[12] Yang Xiaodan, Zhou Lihua, Wang Lizhen. An improved ML-KNN approach based on coupled similarity [C]// Proc of Asia-Pacific Web Conference. Berlin: Springer International Publishing, 2016: 77-89.

[13] 王进，夏翠萍，欧阳卫华，等. Spark 下的并行多标签最近邻算法 [J]. 计算机工程与科学, 2017, 39 (2): 227-235. (Wang Jin, Xia Cuiping, OuYang Weihua, et al. Parallel multi-label K-nearest neighbor algorithm based on Spark [J]. Computer Engineering & Science, 2017, 39 (2): 227-235.)

[14] Zhang Minling, Zhou Zhihua. ML-KNN: a lazy learning approach to multi-label learning [J]. Pattern Recognition, 2007, 40 (7): 2038-2048.

[15] Deng Zhenyun, Zhu Xiaoshu, Cheng Debo, et al. Efficient KNN classification algorithm for big data [J]. Neurocomputing, 2016, 195 (C): 143-148.

[16] ZengYong, Fu Haoming, Zhang Yuping, et al. An improved ML-KNN algorithm by fusing nearest neighbor classification [J]. DEStech Transactions on Computer Science and Engineering, 2016 (aics): 8196.